The Design and Implementation of an Open-Source Hardware Trojan for a 64-bit RISC-V CPU Design

Athanasios Moschos
School of Electrical and Computer Engineering
Georgia Institute of Technology

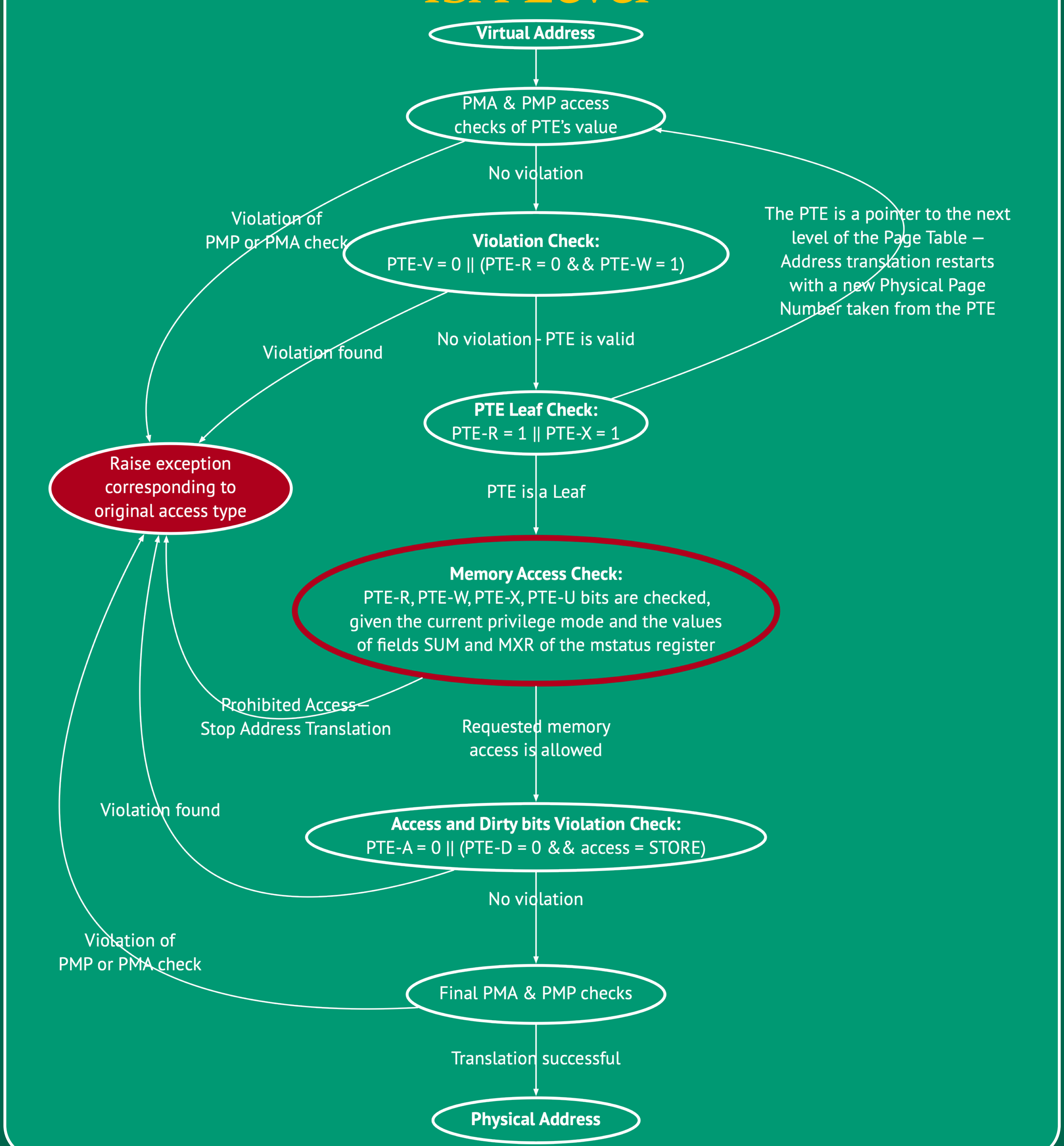
Angelos D. Keromytis
School of Electrical and Computer Engineering
Georgia Institute of Technology

1. MOTIVATION

- Hardware Trojan (HT) paradigms on modern, realistic and complex designs are scarce. Development of corresponding open-source HT-testbeds is important for research conducted on defense mechanisms and detection methods for HT.
- The silicon itself can enable attacks that disable or selectively by-pass fundamental security mechanisms (e.g., memory protection) of modern Central Processing Units (CPUs).
- Overwriting data inside the kernel address space from a user process violates address space isolation, a powerful Operating System security mechanism.
- Current HT detection methods pertain a prohibitive economic cost, are very time consuming and can lead to the destruction of the Device Under Test.

2. ADDRESS TRANSLATION IN RISC-V

ISA Level



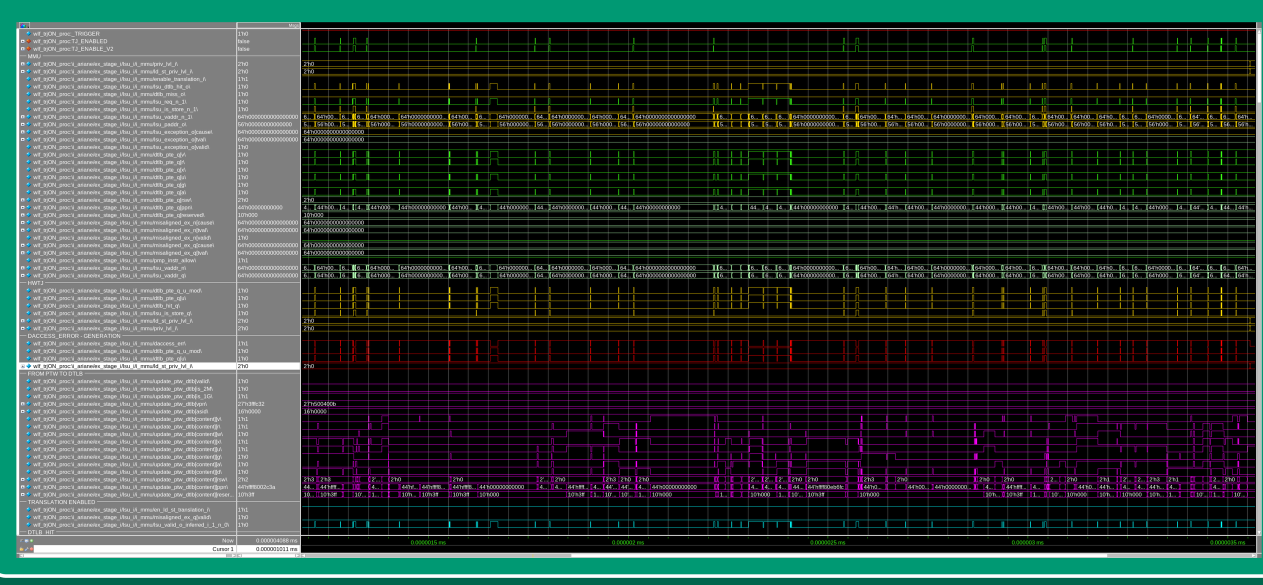
The targeted CPU design is Ariane/CVA6, a 64-bit implementation of the RISC-V Instruction Set Architecture (ISA).

The HT targets memory protected areas of the CPU through the store memory-access mechanism.

Specific memory access checks performed during the address translation process are subverted to enable the attack.

The HT targets the U-bit of the Page Table Entry (PTE). This bit indicates if the processor can access the page while being in User Mode.

Microarchitecture Level

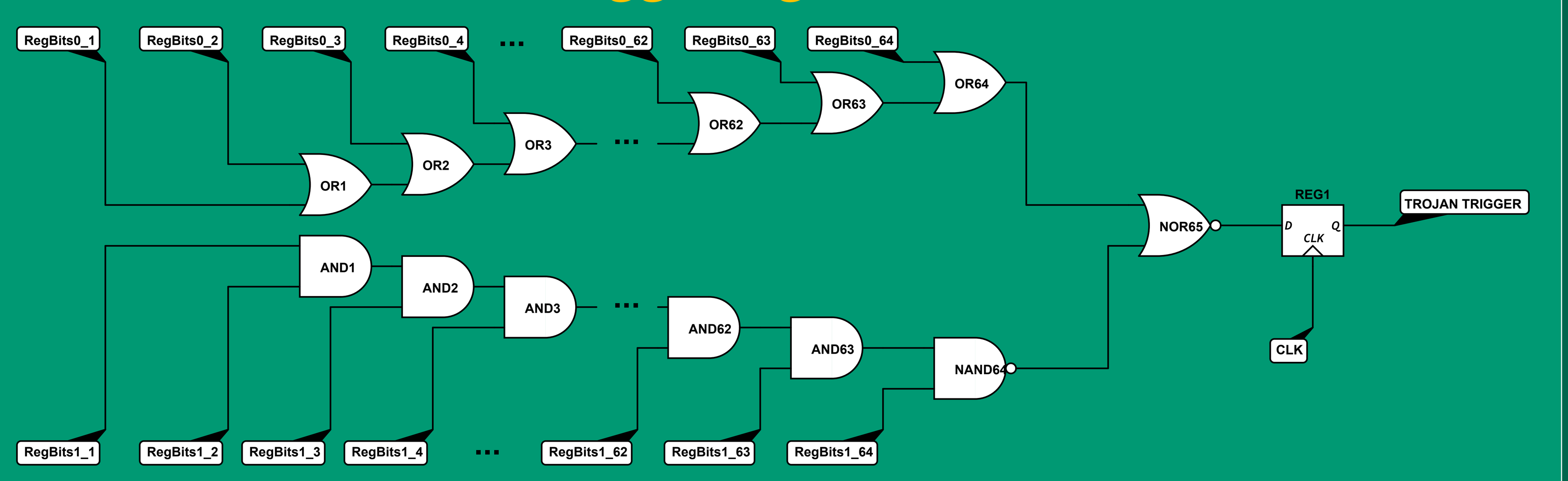


RTL code inspection and real-time measurements from Ariane's internal signal states allow for behavioral profiling of the signals involved in the address translation process.

Candidate signals are then chosen for the trigger and payload circuits.

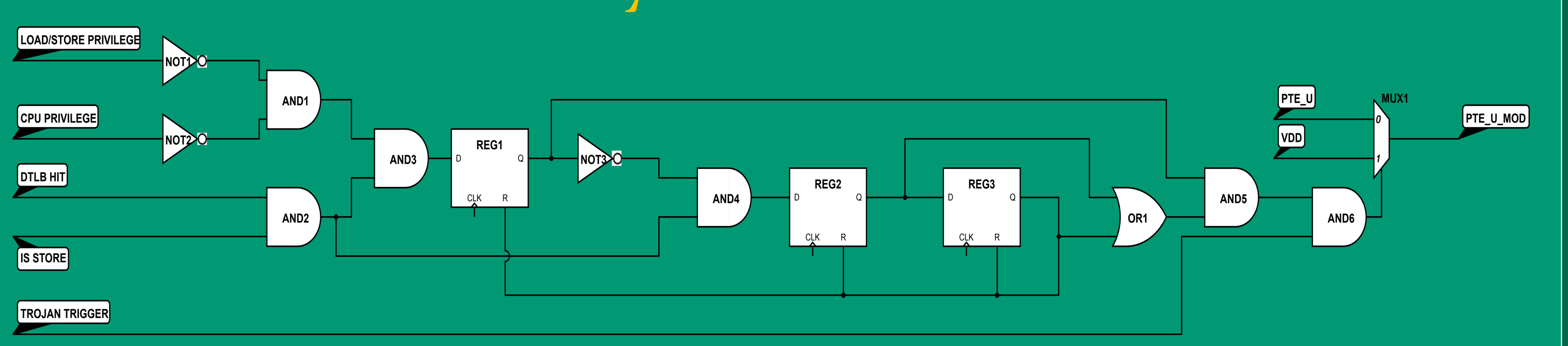
3. DESIGN & IMPLEMENTATION OF ASTRAHAN

Triggering Circuit



The trigger circuit targets Ariane's General-Purpose Registers (GPRs), as it monitors them for specific byte values to assert the trigger signal. Triggering circuit's design is scalable. For the worst-case scenario of a 128-bit magic value, it requires only 130 logic gates.

Payload Circuit



The payload circuit targets Ariane's Memory Management Unit (MMU). Payload circuit design has a minimal footprint, as it requires only 14 logic gates.

FPGA resource utilization of Astrahan's circuits and Ariane's implementation.

Module	LUTs	FFs	LUT Utilization	FF Utilization
*Ariane**	66133	51854	32.45%	12.72%
Register File**	1240	1985	1.87%	3.82%
Trigger Circuit**	26	1	0.00039%	0.000019%
Memory Management Unit**	4271	3388	6.45%	6.53%
Payload Circuit**	4	3	0.0006%	0.000057%

** The utilization percentages referenced here are with respect to the total FPGA resources.*
*** The utilization percentages referenced here are with respect to the total Ariane Implementation.*

4. FUNCTIONALITY & STABILITY EVALUATION

Software Logic for HT-Testing

Linux Kernel Module

```c
/* Kernel Programming */
#include <linux /module.h>
#include <linux /kernel.h>
#include <linux /kern_levels.h>
#include <linux /init.h>
static int vic_var ;
int init_module (void)
{
   vic_var=0xAABBCCDD;
   printk (KERN_ALERT "%px = %pa\n", (void *)&vic_var, (void *)&vic_var);
   return 0;
}
void cleanup_module ( void )
{
   printk (KERN_ALERT "%px = %pa\n", (void *)&vic_var, (void *)&vic_var );
   return 0;
}
MODULE_LICENSE( "GPL" ) ;
```

Once loaded by the OS, the Linux Kernel Module (LKM) assigns a value to a 64-bit variable defined inside the kernel address space.

User Process

```c
/* "Malicious" User Process*/
int main(){
   u_int64_t * ptr;
   // Get the address from stdin
   printf("Memory address to be altered:");
   scanf ("%p", &ptr);

   // Trigger the Trojan
   asm ("addi a5, zero,1\n\t" \
     "sll a5, a5, 0x20\n\t" \
     "lui a6, 0xfffff\n\t" \
     "sll a6, a6, 0x1SSS4\n\t" \
     "xor a6, a6, a5\n\t" \
     "lui a7, 0xfffff \n\t" \
     "sll a7, a7, 0x13\n\t" \
   ) ;
   // Change the value stored in the address.
   *ptr = 0xAAAABBBB;
   return 0;
}
```

The user process initially activates the HT and then alters the value of the variable defined by the LKM. This is not normally allowed by the OS.

Functional testing on Ariane's trojaned RTL code using the RISC-V Assembly unit tests (*Assembly Commands*, Atomic Memory Operations, *Floating-Point and Multiplication).* Tests were executed error-free using Mentor's QuestaSim.

CPU stability evaluation through interleaved real-time execution of *Dhrystone* benchmark and *RAMSpeed* program on the FPGA implemented trojan design.

Complete simulation of the Linux boot-up process in both trojan and trojan-free Ariane. The following internal signaling comparison proved that trojan insertion leaves all the internal signals unaffected.

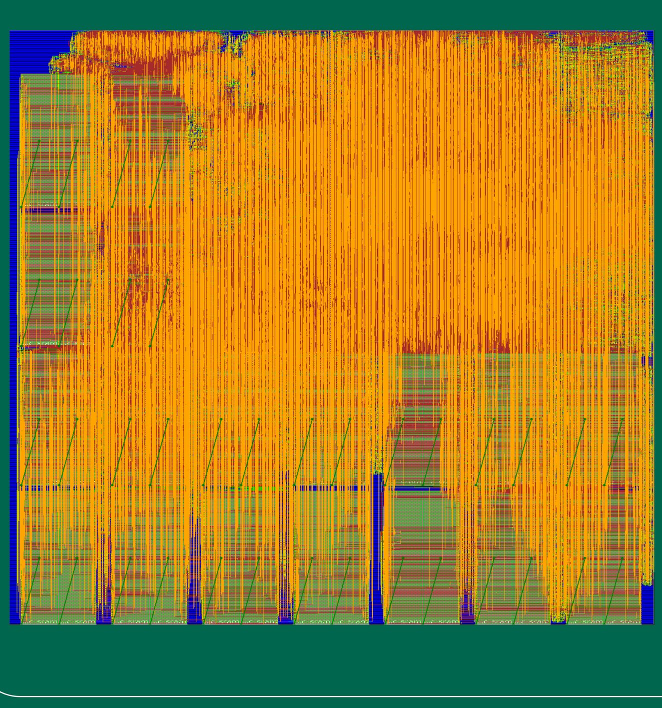
5. ASTRAHAN VS A2

Trojan Name	Gate Number	Host Module	Operational Flexibility	Activation Status	ASIC Feasibility	Open-Source Design
A2	1 & 91	OR1200 (Open RISC)	High	Check needed	Feasible	Yes
Astrahan	144***	Ariane (RISC-V)	Very High	Check not needed	Feasible	Yes

- Open-RISC vs RISC-V ISA.
- Equivalent number of gates for purely digital trojan implementation.
- Astrahan's "context switch safe" design provides very high operability during attack time.
- Astrahan's design requires no prior status activation validation during attack time.
- Astrahan's design provides the scalability required during a fabrication time attack.
- Astrahan's design is open-sourced to the community just like A2's design.

**** The number of gates for the worst-case scenario, where an attacker chooses to use a 128-bit value as the triggering input.*

6. THE ASIC SCENARIO



Astrahan's low-footprint and scalable design alleviates insertion during fabrication as:

- The GPRs and the number of bits utilized by the triggering circuit, can be chosen according to the GPRs' placement and the layout's overall spatial options.
- The time sensitive path of the triggering signal is a multicycle-path, as a page table walk is initiated every time the user process accesses a kernel space virtual address. Thus, Astrahan's design can cope with demanding layouts where the payload and triggering circuit need to be placed far apart.
- Astrahan's design uses a minimum number of flip-flops, thus putting minimum stress on the local clock networks and requiring less space for placement.

The overhead of the trojan gates on the original circuit paths can be calculated by exporting in test-benches only the layout part of concern for accurate RC parasitic extractions. This enables optimal choices for the gate-targets, the trojan gates placement and the routing metals.

7. ONGOING RESEARCH

- Insertion of Astrahan in Ariane's finalized layout using ASIC tools, to showcase the HT's scalability, feasibility and low-footprint design.
- Evaluation of different HT detection methods efficacy with Astrahan's FPGA implementation.
- Further research regarding the identification of points of interest in finalized complex layouts for trojan insertion. Astrahan will be used as a prime example.

ACKNOWLEDGEMENTS

This work is supported by the "Dormant Hardware Trojan Detection Using Back-Scattering Side Channels" project under ONR Contract #N00014-19-1-2287

This is an Open-Source project. Code and examples can be found in: https://github.com/0ena/riscv-hw-trojans

Contact Details:
A. Moschos
(email: amoschos@gatech.edu)
A. D. Keromytis
(email: angelos@gatech.edu)

COEUS Center,
School of Electrical and Computer Engineering,
Georgia Institute of Technology